Mesterséges Intelligenciák féléves feladat

A6NQW1 - Nehéz

2022/23 I. félév

A feladat implementációját – azaz a forráskódot, és a lezáró dokumentumot az `aitflew@uni-miskolc.hu` e-mail címre kell elküldeni, majd azt a 13. és 14. héten a gyakorlatokon kell megvédeni. A kész feladat leadásához a GitHub javasolt, de nem kötelező. Az e-mail tárgya és a küldő jól beazonosítható legyen, ez vonatkozik a GitHub felhasználóra is.

Tetszőleges nyelv, keretrendszer, technológia választható. Az egyetlen megkötés, hogy nem lehet olyan könyvtárat használni, amely tartalmazza a feladat modelljét és/vagy a megoldó algoritmusokat.

A plusz feladatok elvégzésével magasabb érdemjegyet lehet elérni.

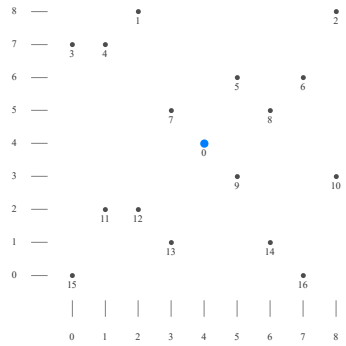
Probléma: Vehicle Routing Problem

A Vehicle Routing Problem (VRP), a Traveling Salesman Problem (TSP) általánosítása. Az ügynök helyett több „futár" megy egy bázisból a városokba. Fontos, hogy a városok mindegyikében egy, és csakis egy futárnak kell járnia. A bázis az egyetlen pont, amelyet több futár érinthet. Mivel a bázis ez esetben különbözik a többi várostól, ezért a modellben külön kell kezelni.

A kereső algoritmusban használt n-opt (a gyakorlatokon a 2-opt került bemutatásra) operátor kétféle kell, hogy legyen. Mivel a futárok számával egyenlő útvonal van, ezért az egyes útvonalakon belül és az útvonalak közt is cserélnünk kell.

Egy lehetséges Vehicle Routing probléma, ahol a depó a 0 számú csomópont és a járművek/futárok száma 4. Ennek egy leírása Python nyelven:

```
[(456, 320), # location 0 - the depot
(228, 0),    # location 1
(912, 0),    # location 2
(0, 80),     # location 3
(114, 80),   # location 4
(570, 160),  # location 5
(798, 160),  # location 6
(342, 240),  # location 7
(684, 240),  # location 8
(570, 400),  # location 9
(912, 400),  # location 10
(114, 480),  # location 11
(228, 480),  # location 12
(342, 560),  # location 13
(684, 560),  # location 14
```



1. ábra: A probléma grafikusan ábrázolva

```
(0, 640),    # location 15
(798, 640)]  # location 16
```

Ekkor a városok közti távokat Manhattan távolságként adjuk meg. (x_1, y_1) és (x_2, y_2) Manhattan távolsága: $|x_1 - x_2| + |y_1 - y_2|$.

A probléma egyik lehetséges megoldása:

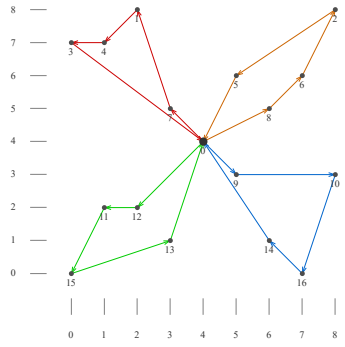
```
Route for vehicle 0:
 0 -> 8 -> 6 -> 2 -> 5 -> 0
Distance of route: 1552m

Route for vehicle 1:
 0 -> 7 -> 1 -> 4 -> 3 -> 0
Distance of route: 1552m

Route for vehicle 2:
 0 -> 9 -> 10 -> 16 -> 14 -> 0
Distance of route: 1552m

Route for vehicle 3:
 0 -> 12 -> 11 -> 15 -> 13 -> 0
Distance of route: 1552m

Total distance: 6208m
```



2. ábra: A megoldás grafikusan ábrázolva

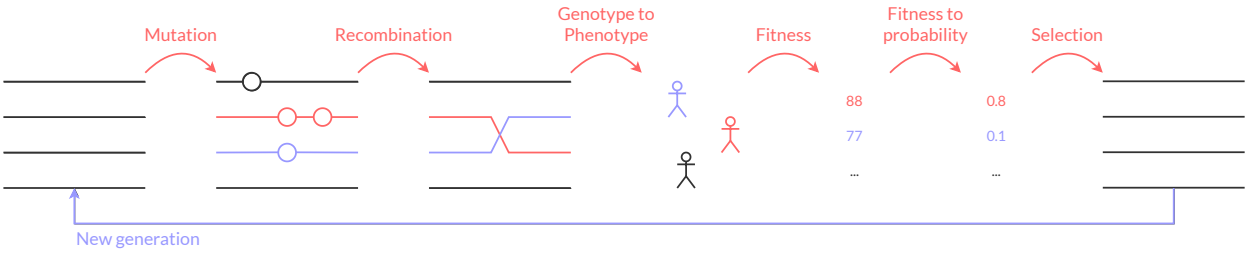
Ezeket a feladatokat inicializálja valamilyen véletlen szám generátorral! Legyen különböző méretűek:

- a városok száma legyen 10, 20, 50, 100, 200, 500,
- a futárok száma legyen 1 (TSP), 2, 4, 5, nagyobb feladatok esetén (ahol a városok száma legalább 50): 10, 20.

A legenerált feladatok legyenek perzisztensen tárolva (vagy seedhez kötéssel procedurálisan generálva).

Algoritmus: Genetikus Algoritmus

A genetikus algoritmusokat gyakran használjuk nevezetes problémák megoldására. Naív természeti megfigyeléseken alapszik. Egy kromoszómákból álló populáción az alábbi műveleteket végezzük iteratívan:



1. Mutáció
2. Rekombináció
3. Genotípus →Fenotípus átalakítás
4. Fitness érték számítása
5. Túlélési valószínűség számítása
6. Túléléssel új generáció képzése

Az egyes kromoszómák a feladat lehetséges megoldásait tartalmazzák (értékek vektora). A mutáció valamilyen elemi (szomszédsági) művelet, melyel az egyes egyedeket (kromoszómákat) alakítjuk. A rekombináció során az egyedeket keresztezzük valamilyen módon, ezzel megtartva a jó tulajdonságokat. A genotípusból fenotípusba átalakítás során az egyes kromoszómáknak jelentést adunk (itt válik a feladat megoldásává). A fenotípus alapján kiszámoljuk a célfüggvényt, ez adja majd az egyed fitness értékét. A fitness értéket felhasználjuk arra, hogy egy valószínűséget számoljunk, ami azt adja meg, hogy az egyed milyen eséllyel éli meg a következő generációt. A kovetkező generáció a túlélő kromoszómákból áll (esetlegesen kívülről bekerülhetnek új, random elemek).

Az egyedek száma egy generációban lehet konstans és változó is. A mutáció és rakombináció során felül írhatjuk az egyes egyedeket, de hozzá is adhatjuk az eddigi elemekhez az újakat.

A mutációnak sokféle változata lehetséges. Ez lehet egyszerre többféle:

- n-opt
- Az egyes, véletlenszerűen választott szegmensek eltolása
- Az egyes, véletlenszerűen választott szegmensek megfordítása
- TSP és VRP esetén a mohó algoritmus alapján egy szegmens rendezése
- VRP esetén két útvonal közti csere
- Flow-shop esetén egy adott szegmens Palmer-index [1] alapján nem növekvő sorrend alapján való rendezése
- Egy véletlenszerűen választott szegmens javítása egy lokális keresővel.

Ezeket egyszerre is használhatjuk, ahol valamilyen valószínűséggel választjuk ki az egyes operátorokat.

A keresztezés szintén lehet véletlenszerű és/vagy valamilyen heurisztikán alapuló. A keresztezendő egyedek kiválasztása is alapulhat heurisztikán és/vagy valamilyen véletlen szám generátoron.

A fitness érték számítása problémafüggő.

A túlélési valószínűség számítása lehet relatív:

[1]

$$I_j = -\sum_{j=1}^{m}\left(\frac{m-(2j-1)}{2}p_{i,j}\right)$$

- j - gép indexe
- m - gépek száma
- $p_{i,j}$ - i-edik munka munkaideje a j gépen

$$P_i = \frac{f_i}{\sum_j f_j},\ f_i \geq 0$$

Lehet rendezés után valamilyen P_c, $0 < P_c < 1$ konstans megadásával. A különbség számítása lehet:

$$\begin{aligned} P_1 &= P_c \\ P_2 &= (1 - P_c) \cdot P_c \\ P_3 &= (1 - P_c)^2 \cdot P_c \\ P_{n-1} &= (1 - P_c)^{n-2} \cdot P_c \\ P_n &= (1 - P_c)^{n-1}, \end{aligned}$$

ahol a generáció egyedeinek száma n, a legjobb fitness értékű egyed az első, a legrosszabb az n-edik.

A túlélésbe bevehetjük az egyed egyedek „életkorát" is.

Az egyes paramétereket, mint a P_c, mutáció, rekombináció operátorainak valószínűségét nem feltétlen kell konstansként kezelni. Valamilyen stratégiával változhatnak. Ehhez érdemes megvizsgálni a Szimulált Hűtés hűtési stratégiáit.

Plusz feladat

Vizsgálja meg, hogy a populáció- és generációszám hogyan hat az eredményre és futási időre! Mikor érjük el azt a pontot, amikor már nem érünk el jobb eredményt új generációk bevezetésével.

Vizsgálja meg, hogy több mutációs és keresztezési operátor bevezetésével hogyan változik az eredmény!

Vizsgálja meg, hogy az egyes paraméterek időtől és/vagy fitnesstől függően, valamilyen stratégia alapján számítása hogyan hat az eredményre!

Az algoritmus végét egy ún. leállási feltétel vizsgálatával érjük el. Ez lehet egy adott generációszám elérése, egy előre meghatározott hőmérséklet elérése, valamennyi generáció eltelte úgy, hogy nem javult az eredmény, vagy ezeknek valamilyen kombinációja. Vessen össze ezekből legalább kettőt, vagy valamilyen kombinációikat. Hogyan hat ez az eredményre és a futási időre?

Készítsen egy alkalmazást, ahol a feladat megadásával elkészíti autómatikusan a megoldást és ezt valahogy megjeleníti (térkép, Gantt diagram).

A feladat lezárása

A féléves feladat lezárásaként egy dokumentumot kell elkészíteni, melyben szerepelnek a megoldás lépései, a választott nyelv, technológia, könyvtárak, keretrendszerek. Esetlegesen, a külön irodalomkutatás eredményeit is tartalmazza.

Minden plusz munka javítja a kapható érdemjegyet. A könnyű feladat csak kivételes esetben érhet el hármasnál jobb jegyet.